A Data Platform for the Highway Traffic Data

Rizwan Mian, Hamoun Ghanbari, Saeed Zareian, Mark Shtern, Marin Litoiu.
School of Information Technology
York University
Toronto, Canada
{rmian@yorku.ca, hamoun.gh@gmail.com, zareian@yorku.ca, shtern@cse.yorku.ca, litoiu@yorku.ca}

Abstract—Both short and long term information of the transportation network is needed by commuters and planners. In order to obtain this information, there is a pressing need to consolidate, mine and analyze data collected from multiple sources. To enable these activities under a single umbrella, we propose a data platform in this position paper that transforms data into information. Finally, we discuss the research challenges facing our platform.

Keywords—Transportation; Big Data; Hadoop; Data Management.

I. INTRODUCTION

Traffic congestions and jams lead to a spectrum of problems including wasting time and energy at one end, and air pollution and mental stress on the other. Often, it is too late to “build” our way out of congestion. Nonetheless, we can explore multiple ways to efficiently utilize existing infrastructure. In order to improve utilization, a better understanding of the transportation structure is needed. Researchers have modeled different aspects of transportation using travel surveys, fluid flow model or game theory [1]. With the emergence of new data sources, such as traffic sensors, cameras, GPS-devices and cell phones, opportunities have emerged for near real-time data analytics\(^1\) and mining. We collectively call data originating from different sources as traffic data.

The velocity and magnitude of data emanation varies across sources. For example, loop detector sensors\(^2\) embedded in the roads of Greater Toronto Area (GTA) collect data at every 20-second intervals. Meanwhile, the social activity varies during the day. The data exists in a variety of forms including numerical, textual and visual. The data is collected over years, and is voluminous in size. Managing and mining this data is truly a big data problem. We see management and mining of traffic data as two related but different problems. Both are needed, but address different research questions.

Examples of the research questions for a data management platform are:

- How to create a data platform that provides real-time ingestion of data from multiple sources?
- How to setup a data platform that can dynamically scale by varying the number of data nodes while in service?
- How to develop such a platform in a cloud? How to use temporary Virtual Machines (VMs) in a cloud effectively?
- What provisioning techniques to employ to handle varying load?
- How to manage data platform between private and public clouds? Does it make sense to divide the platform over multiple clouds?
- How to provide cost-effective data transfer between private and public clouds on a continual basis?

Meanwhile, examples of the research questions for a traffic engineer using data mining techniques are:

- How does congestion in highways evolve over time?
- What factors (weather, sports, news etc.) lead to unexpected congestion?
- Are prediction models for congestion built using sensor and camera readings more accurate than existing models based on publicly available data such as weather and congestion history?
- How would commuters feel about a rise in toll charges on the highways? Would it improve traffic flow?
- How would the traffic volume change in the next five years?

In this position paper, we focus on the data management platform and share our vision. We propose a conceptual architecture and suggest implementation of components based on existing tools in the Hadoop ecosystem [2].

The rest of the paper is organized as follows. Section II presents a motivational analytical study for the proposed data platform. Section III presents different data sources, requirements and the platform architecture. Section IV discusses research challenges and opportunities. Section V discusses related work, and Section VI concludes the paper.

II. TRANSPORTATION ANALYTICS CASE STUDY

The traffic congestion is a serious problem in GTA. We perform an analytical study to shed light on “what are the congestion points in GTA for each weekday and time of the day?” We visualize the speed for each day of the week for the entire data set, and show the congestion for interesting

---

\(^1\) We use the term data analytics loosely to include simple and advanced analytical activities such as extracting metrics and building prediction models, respectively.

\(^2\) Loop detector are sensors embedded in the highways.
intervals on Tuesday. Tuesday represents a typical weekday, and the patterns are similar for the remaining weekdays.

Our analysis is conducted on the data collected from the sensors embedded in the highways of GTA. In the XML format, the size of data is 300 GB for the summer of July 2013. We conduct the study using a traditional analytical tool, namely Matlab. The data is aggregated over an hour interval for each weekday of the data set, and is then analyzed by Matlab.

In Matlab, each sensor location is flagged congested when the aggregated speed readings of its corresponding sensor drops by 30% of the maximum speed readings for that sensor. We then visualize the congested sensor locations using a scatter-plot overlapped on the actual map of GTA. Each congestion point is represented by a red dot point in the plot. The axes of the plots are the latitude and longitude.

Figures 1 to 4 show the outcome of the visualizations. Figure 1 shows that the highways are mostly clear at midnight as expected. We do see a few congestion points, and speculate that congestion exists on these points due to construction work. Figure 2 and Figure 4 show that the highways are congested during the morning and evening rush hours as expected. Interestingly, we see some congestion on the bottom left corner Figure 3. This needs further investigation.

This analysis has been performed over sensor data set whose raw size is 300 GB for three months. It becomes 6TB for the duration of 5 years. It will take substantial computational power to extract, transform and prepare this data for processing which is beyond the capacity of a single server.

With years of sensor data available, opportunities exist for studying changes in congestion points and traffic patterns for each season across the years. This is a much needed exercise for transportation planning and optimizing existing transportation infrastructure. To enable this exercise, extensive computational power is need to explore the data and build comprehensive models on top of it.

Sensor data is only one source of information. Presently, there are multiple sources of traffic data available that differ in the structure and type of data they provide as stated in Table 1. No one source provides correct information at all times, and there is a need to consolidate data from multiple sources to get an accurate picture [3]. Storing and processing data from multiple sources increases the data size and the expectations from a data platform. Big Data methodology and tools are clearly needed to provide such a platform. They are discussed in the next section.
III. DATA MANAGEMENT PLATFORM

Different parties are interested in different aspects of the traffic data. We assume that the traffic analysts need the following types of operations over traffic data in the data management platform:

- Extract standard metrics from traffic data, such as annual average daily traffic, which is of interest to transportation planners and authorities.
- Ad-hoc query processing over traffic data using SQL-like language, such as generating a list of top 10 congested points.
- Support for online statistics, such as mean-speed on a particular highway “right-now”.
- Support offline data-mining and machine-learning, such as building prediction model for congestion.
- Text search over social media content.
- Facilitate efficient route planning for users.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop detector sensors</td>
<td>Average Speed and Traffic Flow per 20-seconds</td>
<td>Numerical</td>
</tr>
<tr>
<td>Traffic Cameras</td>
<td>Blob of video in stream format</td>
<td>Video</td>
</tr>
<tr>
<td>GPS Devices</td>
<td>Location and speed via cellular network</td>
<td>Numerical</td>
</tr>
<tr>
<td>Incident Reports</td>
<td>Witness reported issues</td>
<td>Audio</td>
</tr>
<tr>
<td>Transport Schedules</td>
<td>Tabular schedules of public transportation</td>
<td>Numerical</td>
</tr>
<tr>
<td>Media Outlets</td>
<td>i.e. radio stations reports</td>
<td>Auto/text</td>
</tr>
<tr>
<td>Social Media</td>
<td>Crowd reported information</td>
<td>Text</td>
</tr>
</tbody>
</table>

The characteristics of traffic data presented in Table 1 and the requirements placed by the traffic analysts result in the following requirements on our data platform:

- Highly elastic and scalable to handle constantly increasing size of traffic data, and varying number of users.
- High write throughput to store data generated by multiple sources.
- Efficient range scans (needed to support analytics).
- Robust against failures of a few data nodes.
- Best-effort failure recovery. That is, loss of some data is tolerable to the users.
- Almost high availability for data ingestion (missing some data in downtime is permitted).
- The platform is available for operations most of the time but not all the time and is non-mission critical.
- Ability to execute online analytical jobs.

We consider the following as non-requirements:

1. Full ACID guarantees
2. Frequent updates to existing data
3. Zero-down time in case of a single node failure

We propose the following platform to satisfy the above requirements in Figure 5. We discuss components of the platform below.

- **Storage:** One possibility is to use HBase over HDFS cluster as a storage layer for the engines in the platform.
Underlying HBase is the HDFS cluster, which provides data re-balancing and fault resiliency.

- **Workflow Engine**: Oozie [4] is a candidate for the workflow engine to define data pipelines.

- **Graph Engine**: Titan and Giraph are two candidates to provide a graph database system component.

- **MapReduce Engine**: MapReduce jobs can run directly over HDFS or over HBase.

- **Directory**: HCatalogue is a candidate for data directory that stores the location of different types of data and meta-data. It also provides visibility of data to other components in the platform.

- **Text Engine**: SolrCloud is a candidate for text engine that creates indexes. These indexes can be stored back into the storage.

- **Mining Engine**: We do not have any off-the-shelf candidate for the mining engine of this platform. We plan to develop a mining engine using the machine learning libraries of Mahout.

Next, we state the interfaces of platforms below.

- **Ad-hoc Analytics**: SQL-type search over historical data can be expressed in SQL-like language such as HiveQL.

- **Programming Analytics**: Users develop and submit programs to the data platform. For example, jobs launched in Hadoop are based on user provided programs such as route planning using iterative Graph capabilities of the platform.

- **Text Search**: Users provide terms and to search the textual content in the platform. In addition, they can annotate the terms with conditions and/or ranges.

We expect very large data sizes and varying number of users, and need the platform to adjust to changing demands. Therefore, a management layer based on the MAPE-k loop [5] is an integral part of the platform. The manager monitors the platform, analyzes the current conditions, plans actions to take the platform to a particular state, and executes the corresponding actions. It may acquire, adjust or release resources from the cloud.

We foresee that management will benefit from estimating the platform capacity [6, 7]. Predicting the workload behavior also sets the expectations for execution time and cost. In particular, this enables more reliable guarantees towards any service level agreements (SLA) prior to any workload execution.

IV. RESEARCH ISSUES AND OPPORTUNITIES

We leverage Hadoop ecosystem and we focus on challenges related to scaling, data normalization and representation.

**Vertical vs. horizontal scaling**: There are two types of scaling, namely vertical, where existing VMs are resized [8], or horizontal where VMs are added or removed to the worker pool [9]. Amazon EC2 [10] provides both vertical and horizontal scaling of VMs. In the Amazon’s cloud, a VM can only be resized to a “predefined” VM type, and the VM is unavailable during the resizing operation. On the positive side of resizing, the data need not be moved. Meanwhile, some NoSQL systems, such as Cassandra [11], do not face any downtown and continue to execute workloads during horizontal scaling.

On a separate note, the upper bound on the vertical scaling is fairly low and equal to the “most” powerful host in the cloud. Meanwhile, the upper bound on the horizontal scalability is very large and is typically restricted by the management component of the NoSQL system.

A better approach might be to mix vertical and horizontal scaling. Mixed Scaling increases the complexity in management that needs to keep track of which component benefits from which scaling, and whether any particular type of scaling is of benefit to the overall system.

**Scaling platform components**: Our platform contains multiple components at different levels, and pose an interesting scaling problem. That is, how to scale when a subset or a single component is experiencing load. For example, the load on the text engine suddenly increases near a major sporting event, while the load remains normal at other component. In this case, it is appropriate to scale the components that are experiencing high load. To support this, we need to separate out data storage from engines (e.g. MapReduce engine). The data is then stored in the storage layer, while the engines primarily perform computation. This separation makes it possible to add more VMs containing engine binaries to the system’s fleet. The data is persisted in the storage layer, and the additional VMs are released when the load subsides.

**Data normalization**: There is a wide variety of input data to our platform. One approach is to store the raw data in the storage layer without any filtering or processing, and let the engines treat the data as required. This approach is “fast” and stores real-time data without any delay. However, the size of raw data might be excessively large and the cleaning operation is duplicated at the engines. This is aggravated if there is no “value” in the raw data stored. Instead, the raw data can be filtered, cleaned and aggregated in some way before being stored. This will most certainly reduce the size of the data stored and result in lesser load on the storage layer. The aggregation may also reveal the “current” trend. However, the on-route processing is likely to make some assumptions, for example, replace missing values with today’s average. These assumptions might be valid in some cases, but not in others. Further, finding “gems” in big data is typically a discovery process, and it is difficult to tell in advance if the raw data has any jewels. By only storing the processed data, we lose the opportunity to conduct a discovery exercise later. Where to draw the line between on-route processing and storing raw data is an open research question.

**Data representation**: The system consists of multiple components, and would have some inter-operability at the cost of reduced functionality. For example, Hive can operate directly over HBase tables and do not need to store a copy of the data. However, there can only be “one” index on an HBase
table, and Hive is unable to benefit from creating multiple indexes. As a result, Hive needs to scan more data using MapReduce jobs, which it did not need to in case of multiple indexes. HBase gets around the inability to create multiple indexes for its operations by creating a “nested” index. For example, 10.05am-jones represents a nested key to the record when Jones logged in to some portal. However, Hive is presently unable to benefit from the nested key. The trade-off lies between using a uniform data representation as much as possible at the cost of reduced functionality or using specialized functionality to maximum functionality at the cost of multiplying data redundancy and risk of data inconsistency.

V. RELATED WORKS

Our work proposes a Big Data platform specialized to traffic data. Therefore, we discuss both existing data management systems for traffic data and systems dealing with Big Data in general.

There are about 1,600 loop detector sensors and 200 cameras located in the highways of northern Belgium. The measurements, collected at the frequency of 1m, are stored into a central database in the raw, unprocessed and non-validated form [12]. The California Freeway Performance Measurement System (PeMS) has about 26,000 loop detector sensors collection data at 30s interval into an Oracle database system [13].

The city of Bellevue has about 180 loop detector sensors, and the data captured is available in CSV files at every minute interval. GATI system [14] downloads the traffic data from the Bellevue data server and stores it in a MySQL database system. Hoh et al. [15] and Lo et al. [16] collect traffic data by probing GPS-equipped vehicles, and store it in a Microsoft SQL Server and PostgreSQL respectively.

The above systems usually collect data from a single source, which has structured type, and is stored in a relational database system. The traditional database systems have limitations over horizontal scalability [17]. The above systems display the collected data on a map, estimate travel times, or show current traffic conditions in a web-browser or over cell phones.

In contrast, our data platform will collect data from multiple sources, which have multiple types, and would be stored in a scalable NoSQL layer. The vision is to provide a comprehensive data portal for traffic analysts by offering multiple interfaces. Data from multiple sources can be combined to lead to new insights. For example, it will be possible to study effects of introducing new toll charges on traffic volumes and reaction of travelers using the tolled highways. However, developing efficient and scalable platforms for Big Data is actively being researched [18-21]. Building such a platform is a multi-facet problem, and we provide examples of research in addressing a particular aspect of the problem.

Yu et al. [22] use the Hadoop ecosystem to build an adaptable data mining platform for transportation in a cloud. Borthakur et al. [18] use HBase over HDFS to support real-time data ingestion and analytics, and rely on HDFS for data availability. They find that the consistency semantics of HBase satisfies user’s expectations.

Konstantinou et al. [20] et al. evaluate dynamic scalability of HBase, Cassandra and Riak. They discover that while adding new nodes improve throughput, data rebalancing is only worthy if the system undergoes continuous high load. They provide a platform that explores the scalability of NoSQL databases, while our platform needs to address scalability of both storage and data processing.

Rabkin et al. [23] explores the reasons for the downtime of a Hadoop cluster. They discover misconfiguration to be the biggest reason for failures. Heger [24] presents a methodology to tune a Hadoop cluster for varying workload conditions. Meanwhile, Rao et al. [25] explores the performance issues of Hadoop in heterogeneous clusters and suggest possible ways to address the issues. Rabkin et al. and Rao et al. explores methods to reduce downtown and improve performance, and our work incorporate their ideas in building an autonomic management.

We also see several commercial solutions such as Google3 and Inrix4 mainly focus on providing predefined traffic analytics and reports accessible through dashboards and/or APIs. However, our work focuses on providing a generic platform that enables ad-hoc analytics over traffic data.

VI. CONCLUSION

In this paper, we propose a platform architecture to support analytics over traffic data. The platform includes multiple engines to support various types of analytics and processing ranging from text searching to route planning. Our platform exposes ad-hoc, programmatic and text searching interfaces. These interfaces accommodate both non-programmers and programmers to carry out analytics. We consider scalability and autonomy in the platform from the outset and provide a management layer based on MAPE loop.

ACKNOWLEDGMENT

This research was supported by IBM Centres for Advanced Studies (CAS), the Natural Sciences and Engineering Council of Canada (NSERC) under the Smart Applications on Virtual Infrastructure (SAVI) Research Network, and the Ontario Research Fund for Research Excellence under the Connected Vehicles and Smart Transportation (CVST) project. We acknowledge the contribution of the ONE-ITS platform in providing access to real-time traffic data.

REFERENCES


3 support.google.com/maps/answer/144359?hl=en
4 www.inrix.com/


